树：

树：是一种抽象的数据类型或是实作这种抽象数据类型的数据结构，用来模拟具有树状结构性质的数据集合。它是由n(n>=1)个有限节点组成一个具有层次关系的集合。把它叫做‘树’是因为它看起来像一颗倒挂的树，也就是说它是根朝上，而叶朝下的。

它具有以下的特点：

（1、每个节点有**零个**或**多个**子节点

（2、**没有父节点的节点称为根节点**

（3、每一个非根节点有且只有一个父节点

（4、除了根节点外，每个子节点可以分为多个不相交的子树

树的术语：

节点的度：一个节点含有的子树的个数称为该节点的度

树的度：一颗树中，**最大的节点的度**称为树的度

叶节点或终端节点：度为零的节点

父亲节点或父节点：若一个节点含有子节点，则这个节点称为其子节点的父节点

孩子节点或子节点：一个节点含有的子树的根节点称为该节点的子节点

兄弟节点：具有相同父亲节点的节点称为兄弟节点

节点的层次：从根开始定义起，根为第一层，根的子节点为第二层，以此类推

树的高度或深度：树中节点的最大层次

堂兄弟节点：父节点在同一层的节点互为堂兄弟

节点的祖先：从根到该节点所经分支上的所有节点

子孙：以某节点为根的子树中任一节点都成为该节点的子孙

森林：由m(m>=0)颗互不相交的树的集合称为森林

树的种类：

无序树：树中任意节点的子节点之间没有顺序关系，这种树称为无序树，也成为自由树

有序树：树中任意节点的子节点之间有顺序关系，这种树称为有序树

（1、二叉树：每个节点最多含有两个子树的树称为二叉树

1、完全二叉树：

对于一颗二叉树，假设其深度为d(d>1).除了d层外，其他各层的节点数目均已达到最大值，且d层所有节点从左向右连续地紧密排列，这样的二叉树被称为完全二叉树，其中满二叉树的定义是所有叶节点都在最底层的完全二叉树

2、平衡二叉树：当且仅当任何节点的两颗子树的高度差不大于1的二叉树

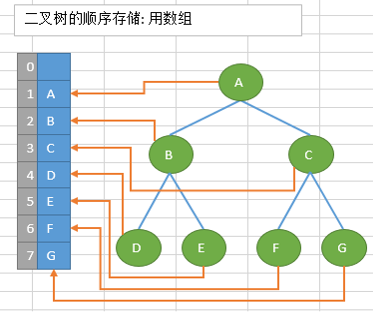
3、排序二叉树：也称二叉搜索树，有序二叉树

霍夫曼树：（用于信息编码）：带权路径最短的二叉树称为霍夫曼树或最优二叉树

B树：一种对读写操作进行优化的自平衡的二叉查找树，能够保持数据有序，拥有多余两个子树

树的存储与表示

顺序存储：将数据结构存储在固定的数组中，然在遍历速度上有一定的优势，但因所占空间比较大，是非主流二叉树。二叉树通常以链式存储



链式存储：

可以存储，缺陷：指针域指针个数不定

解决方案：把多叉树转成二叉树

二叉树：

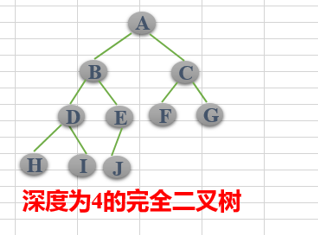
二叉树是每个节点最多有两个子树的树结构，通常子树被称作’左子树‘和’右子树‘

二叉树又可以细分为’完全二叉树‘和’满二叉树‘两种特殊形态

完全二叉树：

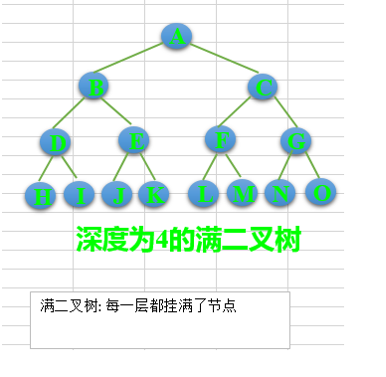
若设二叉树的高度为h,除第h层外，其他各层(1~h-1)的节点数都达到最大个数，第h层有叶子节点，并且叶子节点都是从左到右依次排布，这就是完全二叉树

比如有n层，第一层到第n-1层与满二叉树一样，第n层的最后一个节点前面都挂满了节点



满二叉树：

除了叶节点外每一个节点都有左右子叶且叶子节点都处在最底层的二叉树



二叉树的节点表示：

通过使用Node类中定义三个属性，分别为elem本身的值，还有lchild左孩子和rchild右孩子

Class Node(object):

‘’’节点类’’’

Def \_\_init\_\_(self,elem=-1,lchild=None,rchild=None):

Self.elem=elem

Self.lchild=lchild

Self.rchild=rchild

树的创建：

创建一个树的类，并给一个root根节点，一开始为空，随后添加节点

**class** **Tree(**object**):**

"""树类"""

**def** \_\_init\_\_**(**self**,** root**=None):**

self**.**root **=** root

**def** add**(**self**,** elem**):**

"""为树添加节点"""

node **=** Node**(**elem**)**

#如果树是空的，则对根节点赋值

**if** self**.**root **==** **None:**

self**.**root **=** node

**else:**

queue **=** **[]**

queue**.**append**(**self**.**root**)**

#对已有的节点进行层次遍历

**while** queue**:**

#弹出队列的第一个元素

cur **=** queue**.**pop**(**0**)**

**if** cur**.**lchild **==** **None:**

cur**.**lchild **=** node

**return**

**elif** cur**.**rchild **==** **None:**

cur**.**rchild **=** node

**return**

**else:**

#如果左右子树都不为空，加入队列继续判断

queue**.**append**(**cur**.**lchild**)**

queue**.**append**(**cur**.**rchild**)**